

124



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/874,170	06/04/2001	Vasanth Bala	10003355-1	7644

7590 09/23/2004

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, CO 80527-2400

EXAMINER

PROCTOR, JASON SCOTT

ART UNIT	PAPER NUMBER
----------	--------------

2123

DATE MAILED: 09/23/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary

Application No.

09/874,170

Applicant(s)

BALA ET AL.

Examiner

Jason Proctor

Art Unit

2123

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☐ Responsive to communication(s) filed on ____.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-24 is/are pending in the application.
- 4a) Of the above claim(s) ____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) ____ is/are allowed.
- 6) ☒ Claim(s) 1-24 is/are rejected.
- 7) ☐ Claim(s) ____ is/are objected to.
- 8) ☐ Claim(s) ____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 04 June 2001 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
- Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
- Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. ____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date 8/20/2001.
- 4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. ____.
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other: ____.

Detailed Action

1. Claims 1-24 have been rejected.

Claim Objections

2. Claims 7 and 16 are objected to because of the following informalities: Where a claim sets forth a plurality or elements or steps, each element or step of the claim should be separated by a line intention. See MPEP 608.01(m). Appropriate correction is required.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims 1-3, 6-8, 9-11, 14-19, and 22-24 are rejected under 35 U.S.C. 102(b) as being anticipated by Chernoff et al. US Patent No. 6,000,028.

5. Regarding claim 1, Chernoff et al. discloses a client and server system (Fig. 28; column 34, lines 42-55) wherein

The server includes an application code source (column 34, lines 56-65; Fig. 2, elements 36, 36a, and 17c) and a server code manager coupled to the

Art Unit: 2123

application code source ("background process" of column 10, lines 6-15; column 10, lines 22-37),

a client is coupled to the server (column 34, lines 42-55; Fig. 2, elements 32, 32a, 32b, and 36) and comprises

a CPU for natively executing code segments (column 7, lines 22-32),

code segments are derived from non-native code (column 8, lines 33-42),

the Alpha® microprocessor is known to have a code cache coupled to the CPU

(See Alpha Processor Product Brief), and

a client code manager which requests code segments from a server process, receiving code segments, storing code segments in the cache, executing the code segment using the CPU until the code segment attempts to pass control to a required code segment not stored in the cache, at which point control passes to the client code manager to retrieve the required code segment from a server process, with the CPU continuing execution with the required code segment (column 9, lines 14-28; column 9, line 60 – column 10, line 5; column 10, lines 16-21; column 10, line 66 – column 11, line 14).

6. Regarding claim 2, Chernoff et al. discloses that a background process transforms the client application from a first format to a native binary format compatible with a native instruction set of the CPU of the client (column 10, lines 6-15; column 9, lines 29-47) and a background process parses the translated code into segments

Art Unit: 2123

(column 9, line 60 – column 10, line 15). The server process schedules transactions within and between the run-time and background systems (column 10, lines 54-60).

7. Regarding claim 3, Chernoff et al. discloses that the first format is a native binary format other than the native binary format of the CPU of the client, and the background process transforms the client application from the first format to the native binary format of the CPU of the client (column 10, lines 6-15; column 10, line 66 – column 11, line 14).

8. Regarding claim 6, Chernoff et al. discloses a client code segment manager that requests needed segments from a server process (column 9, line 60 – column 10, line 21; column 10, line 66 – column 11, line 14) and emits the received code segment into the code cache and branches to the received code segment (column 9, lines 29-47).

9. Regarding claim 7, Chernoff et al. discloses that the background process adjusts any branch targets in the received code segment and in the code cache so that the transformed code executes properly (column 10, lines 16-21; column 9, lines 29-47; column 29, lines 27- 39; column 30, lines 54-62).

10. Regarding claim 8, Chernoff et al. discloses the use of an Alpha® processor, which is known in the art to use a cache system to remove old and unneeded code segments from the code cache, replacing older segments with newly received segments when the cache reaches a certain threshold, and optimizing the code segments in the code cache (See "Computer Architecture", page 399-400, "Which Block Should Be Replaced on a Cache Miss?"). Optimization of code segments is achieved by algorithms such as the "Least-recently used" strategy, which attempts to keep frequently needed code segments present in the cache.

Art Unit: 2123

11. Regarding claim 9, Chernoff et al. discloses a networked client and server system (Fig. 28; column 34, lines 42-55) wherein

the server includes an application code source (column 34, lines 56-65; Fig. 2, elements 36, 36a, and 17c) and a server code manager coupled to the application code source ("background process" of column 10, lines 6-15; column 10, lines 22-37),

a server code manager coupled to the application code source which derives native binary code segments in a native execution format required by the client CPUs from the application code source (column 9, lines 29-47; column 9, line 60 – column 10, line 5) and transmits the native binary code segments to clients (column 10, line 66 – column 11, line 14).

12. Regarding claim 10, Chernoff et al. discloses an application code transformation manager for transforming the client application from a first format to the native execution format and a code segment manager for parsing the application into code segments (column 9, line 60 – column 10, line 21).

13. Regarding claim 11, Chernoff et al. discloses that the first format is a native binary format other than the native binary format of the CPU of the client, and the background process transforms the client application from the first format to the native binary format of the CPU of the client (column 9, lines 14-47; column 10, lines 6-15).

Art Unit: 2123

14. Regarding claim 14, Chernoff et al. discloses a networked client and server system (Fig. 28; column 34, lines 42-55) wherein the client comprises

a CPU for natively executing code segments derived from the stored application (column 7, lines 22-32),

code segments are derived from non-native code (column 8, lines 33-42),

the Alpha® microprocessor is known to have a code cache coupled to the CPU for storing code segments (See Alpha Processor Product Brief), and

a run-time system which requests code segments from a server process, receiving code segments, storing code segments in the cache, executing the code segment using the CPU until the code segment attempts to pass control to a required code segment not stored in the cache, at which point control passes to the client code manager to retrieve the required code segment from a server process, with the CPU continuing execution with the required code segment (column 9, lines 14-28; column 9, line 60 – column 10, line 5; column 10, lines 16-21; column 10, line 66 – column 11, line 14).

15. Regarding claim 15, Chernoff et al. discloses that the run-time system is networked (column 34, lines 33-65), coupled to the code cache (column 9, lines 29-47), and requests needed code segments from the server process (column 10, line 66 – column 11, line 14). Chernoff et al. also discloses a code cache linker and manager coupled to the code cache which links the code segment received from the server into

Art Unit: 2123

the code cache and branches to the received code segment in the code cache (column 9, lines 29-47; column 10, lines 16-21).

16. Regarding claim 16, Chernoff et al. discloses that the background process adjusts any branch targets in the received code segment and in the code cache so that the transformed code executes properly (column 10; lines 16-21; column 9, lines 29-47; column 29, lines 27- 39; column 30, lines 54-62).

17. Regarding claim 17, Chernoff et al. discloses the use of an Alpha® processor, which is known in the art to use a cache system to remove old and unneeded code segments from the code cache, replacing older segments with newly received segments when the cache reaches a certain threshold, and optimizing the code segments in the code cache (See "Computer Architecture, page 399-400, "Which Block Should Be Replaced on a Cache Miss?"). Optimization of code segments is achieved by algorithms such as the "Least-recently used" strategy, which attempts to keep frequently needed code segments present in the cache.

18. Regarding claim 18, Chernoff et al. discloses a client and server system (Fig. 28; column 34, lines 42-55) wherein

a client issues a code segment request to a server (column 10, line 66 – column 11, line 14),

a server receives the code segment request from the client (column 10, line 66 – column 11, line 14),

Art Unit: 2123

a background process of the server derives a code segment in a native execution format required by the client from an application code source (column 10, lines 6-14; column 10, line 66 – column 11, line 14; Fig. 2, elements 34, 36b, 36, 36a, and 17c),
transmitting the code segment to the client (column 10, line 66 – column 11, line 14),
the client receives the code segment (column 10, line 66 – column 11, line 14),
the client adjusts branches in the code segment having targets not in a code cache to of the client to cause code segments containing the targets to be requested from the server (column 13, lines 10-30; column 9, line 60 – column 10, line 14),
the code segment is emitted into the code cache and executed natively (column 9, lines 14-47; column 9, line 60 – column 10, line 5).

19. Regarding claim 19, Chernoff et al. discloses that the first format is a native binary format other than the native binary format of the CPU of the client, and the background process transforms the client application from the first format to the native binary format of the CPU of the client (column 10, lines 6-15; column 10, line 66 – column 11, line 14).

20. Regarding claim 22, Chernoff et al. discloses a method to execute a code segment from cache (column 9, lines 14-28) wherein

A client executes a branch in the first code segment that seeks to branch to a second code segment not in the code cache and issuing a request for the

Art Unit: 2123

second code segment to the server (column 29, line 27 – column 30, line 5; column 10, lines 6-14; column 10, lines 22-37),

a server receives the code segment request from the client (column 10, line 66 – column 11, line 14),

a background process of the server derives a code segment in a native execution format required by the client from an application code source (column 10, lines 6-14; column 10, line 66 – column 11, line 14; Fig. 2, elements 34, 36b, 36, 36a, and 17c),

transmitting the code segment to the client (column 10, line 66 – column 11, line 14),

the client receives the code segment (column 10, line 66 – column 11, line 14),

the client adjusts branches in the first code segment that need to branch into the second code segment to branch to appropriate locations within the second code segment (column 13, lines 10-30; column 9, line 60 – column 10, line 14),

the client adjusts branches in the second code segment having targets in the first code segment to branch to the appropriate locations within the first code segment (column 13, lines 10-30; column 9, line 60 – column 10, line 14),

the client adjusts branches in the code segment having targets not in a code cache to of the client to cause code segments containing the targets to be requested from the server (column 13, lines 10-30; column 9, line 60 – column 10, line 14), and

the code segment is emitted into the code cache and executed natively (column 9, lines 14-47; column 9, line 60 – column 10, line 5).

21. Regarding claim 23, Chernoff et al. discloses a client and server software product (Fig. 28; column 34, lines 42-55) wherein

a client issues a code segment request to a server (column 10, line 66 – column 11, line 14),

a server receives the code segment request from the client (column 10, line 66 – column 11, line 14),

a background process of the server derives a code segment in a native execution format required by the client from an application code source (column 10, lines 6-14; column 10, line 66 – column 11, line 14; Fig. 2, elements 34, 36b, 36, 36a, and 17c),

transmitting the code segment to the client (column 10, line 66 – column 11, line 14),

the client receives the code segment (column 10, line 66 – column 11, line 14),

the client adjusts branches in the code segment having targets not in a code cache to of the client to cause code segments containing the targets to be requested from the server (column 13, lines 10-30; column 9, line 60 – column 10, line 14),

the code segment is emitted into the code cache and executed natively (column 9, lines 14-47; column 9, line 60 – column 10, line 5).

22. Regarding claim 24, Chernoff et al. discloses

A client executes a branch in the first code segment that seeks to branch to a second code segment not in the code cache and issuing a request for the second code segment to the server (column 29, line 27 – column 30, line 5; column 10, lines 6-14; column 10, lines 22-37),

a background process of the server derives a code segment in a native execution format required by the client from an application code source (column 10, lines 6-14; column 10, line 66 – column 11, line 14; Fig. 2, elements 34, 36b, 36, 36a, and 17c),

transmitting the code segment to the client (column 10, line 66 – column 11, line 14),

the client receives the code segment (column 10, line 66 – column 11, line 14),

the client adjusts branches in the first code segment that need to branch into the second code segment to branch to appropriate locations within the second code segment (column 13, lines 10-30; column 9, line 60 – column 10, line 14),

the client adjusts branches in the second code segment having targets in the first code segment to branch to the appropriate locations within the first code segment (column 13, lines 10-30; column 9, line 60 – column 10, line 14),

the client adjusts branches in the code segment having targets not in a code cache to of the client to cause code segments containing the targets to be

Art Unit: 2123

requested from the server (column 13, lines 10-30; column 9, line 60 – column 10, line 14), and

the code segment is emitted into the code cache and executed natively (column 9, lines 14-47; column 9, line 60 – column 10, line 5).

Claim Rejections - 35 USC § 103

23. The following is a quotation of 35 U.S.C. §103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

24. Claims 4, 12, and 20 are rejected under 35 U.S.C. §103(a) as being unpatentable over Chernoff et al. as applied to claims 2, 10, and 18 above.

25. Regarding claims 4, 12, and 20, Chernoff et al. does not disclose transforming code from a source code text format of a programming language to a native binary format of the CPU of the client. A device which performs this function is known in the art as a cross-compiler (See Microsoft Computer Dictionary, Fifth Edition). It would have been obvious to a person of ordinary skill in the art at the time of applicant's invention to use techniques such as cross-compiling to better realize the code transformation abilities in the invention of Chernoff et al. The cross-compiler could be implemented similarly to the native binary translation background process of Chernoff et al., residing on the server with access to the client application code source and using

Art Unit: 2123

cross-compiling techniques to produce transformed code segments for the server to transmit to the client.

26. Claims 5, 13, and 21 are rejected under 35 U.S.C. §103(a) as being unpatentable over Chernoff et al. as applied to claims 2, 10, and 18 above and further in view of "Distributed Virtual Machines: A System Architecture for Network Computer" by Sirer, Grimm, Bershad, Gergory, and McDirmid, referred to hereafter as Sirer et al.

27. Regarding claims 5, 13, and 21, Chernoff et al. does not disclose transforming code from a virtual machine format using a just-in-time compiler that compiles and links the client application into a native binary format of the CPU of the client.

28. Sirer et al. discloses a distributed virtual machine wherein a compilation server compiles code into native code for virtual machines that do not support local compilation or interpretation (page 2, lines 25-33; Figure 1). It would have been obvious to a person of ordinary skill in the art at the time of applicant's invention to make use of cross platform code execution techniques, such as compiling virtual machine code into native code, to better realize the code transformation abilities in the invention of Chernoff et al. The compilation server could be implemented similarly to the native binary translation background process of Chernoff et al., residing on the server with access to the client application code source and using just-in-time compilation to produce transformed code segments for the server to transmit to the client.

Art Unit: 2123

Conclusion

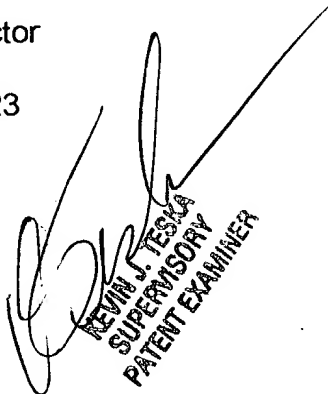
Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jason Proctor whose telephone number is (703) 305-0542 or (571) 272-3713 beginning in October 2004. The examiner can normally be reached on 8am-4pm M-F.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kevin J Teska can be reached on (703) 305-9704 or (571) 272-3716 beginning in October 2004. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Jason Proctor
Examiner
Art Unit 2123

jsp


KEVIN J. TESKA
SUPERVISORY
PATENT EXAMINER